Computing Optimal Hypotheses Efficiently for Boosting

(Draft as of Feb. 20, 2000)

Jun Sese
University of Tokyo
sesejun@gi.k.u-tokyo.ac.jp

Shinichi Morishita
University of Tokyo
moris@k.u-tokyo.ac.jp

Abstract

This paper sheds light on a strong connection between AdaBoost and several optimization algorithms for data mining. AdaBoost has been the subject of much interests as an effective methodology for classification task. AdaBoost repeatedly generates one hypothesis in each round, and finally it is able to make a highly accurate prediction by taking a weighted majority vote on the resulting hypotheses. Freund and Schapire have remarked that the use of simple hypotheses such as single-test decision trees instead of huge trees would be promising for achieving high accuracy and avoiding overfitting to the training data. One major drawback of this approach however is that accuracies of simple hypotheses may not always be high, hence demanding an efficient way of computing the most accurate simple hypothesis. In this paper, we consider several classes of simple but expressive hypotheses such as ranges and regions for numeric attributes, subsets of categorical values, and conjunctions of Boolean tests. For each class, we develop an efficient algorithm for choosing the optimal hypothesis.

1 Introduction

Classification has been a prominent subject of study in the machine learning and data mining literature. Let $\mathbf{x}_i$ denote a vector of values for attributes, which is usually called a *record* or a *tuple* in the database community. Let y_i denote the objective Boolean value that is either 1 or 0. We call a record $(\mathbf{x}_i, y_i)$ *positive* (resp. *negative*) if $y_i = 1$ ($y_i = 0$). Given $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ as a training dataset, classification aims at deriving rules that are able to predict the objective value of y from $\mathbf{x}$ with a high probability.

For classification problems, decision trees are used mostly in practical applications. Recently, to further improve the power of existing classifiers, boosting techniques have received much interest among the machine learning and data mining communities [13]. A classifier is called a *weak hypothesis* if its predication accuracy regarding the training dataset is at least better than 1/2. A boosting algorithm tries to generate some weak hypotheses so that it is able to make a highly accurate prediction by combining those weak hypotheses. There have been many proposals for such boosting algorithms [13, 8]. Freund and Schapire presented the most successful algorithm, named "AdaBoost", that solved many of the practical difficulties of the earlier boosting algorithms [10].

2 AdaBoost

The key idea behind AdaBoost is to maintain the record weights in the training dataset. AdaBoost assumes the existence of a *weak learner* that is able to output a weak hypothesis in a finite number of steps. The aim of this paper is to propose efficient weak learners, but for the purpose of

explanation, we continue the discussion by assuming weak learners. In each iteration AdaBoost calls on a weak learner to generate one weak hypothesis by considering the weighted records as the training dataset and updates the weights of the records to force the next call of the weak learner focus on the mis-predicted records. In this way, we prepare a set of voters with different characteristics. In the final step, we define the weight of each voter according to its prediction accuracy in the training dataset, and we generate the final hypothesis using a weighted majority vote.

Pseudo-code

We now present a pseudo-code for AdaBoost. First, the inputs to AdaBoost are a training dataset $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, the initial weights $w_i^1 = 1/N$ $(i = 1, \ldots, N)$, a weak learner named `WeakLearn`, and the integer T specifying number of iterations. AdaBoost repeats the following three steps for each $t = 1, 2, \ldots, T$:

1. Calculate the distribution p_i^t of each record $(\mathbf{x}_i, y_i)$ by normalizing weights w_i^t; namely,
$$p_i^t = \frac{w_i^t}{\sum_{i=1}^N w_i^t}.$$

2. Invoke `WeakLearn` to produce such a weak hypothesis $h_t : \mathbf{X} \to \{1, 0\}$ that the error ϵ_t of h_t is less than $1/2$, where ϵ_t is defined:
$$\epsilon_t = \sum_{i=1}^N p_i^t \ |h_t(\mathbf{x}_i) - y_i|.$$
Observe that $|h_t(\mathbf{x}_i) - y_i|$ is 1 if h_t mis-predicts the objective value of $(\mathbf{x}_i, y_i)$; namely, $h_t(\mathbf{x}_i) \neq y_i$. Otherwise, $|h_t(\mathbf{x}_i) - y_i| = 0$.

3. Set $\beta_t = \epsilon_t/(1 - \epsilon_t)$. Note that $\beta_t < 1$ since $\epsilon_t < 1/2$. We then set the

new weight w_i^{t+1} for each $i = 1, \ldots, N$ to be

$$w_i^{t+1} = w_i^t \ \beta_t^{1-|h_t(\mathbf{x}_i) - y_i|}.$$

Observe that $1 - |h_t(\mathbf{x}_i) - y_i|$ is 0 if $h_t(\mathbf{x}_i) \neq y_i$, and is 0 otherwise. Thus if $h_t(\mathbf{x}_i) \neq y_i$, $\beta_t^{1-|h_t(\mathbf{x}_i) - y_i|} = 1$, and hence the weight of $(\mathbf{x}_i, y_i)$ is unchanged ($w_i^{t+1} = w_i^t$). On the other hand, if $h_t(\mathbf{x}_i) = y_i$, the weight of $(\mathbf{x}_i, y_i)$ is decreased because $w_i^{t+1} = \beta_t w_i^t$. Observe that the weights of incorrectly predicted records are relatively increased so that the weak learner can focus on these "hard" records in the next step.

Lastly, AdaBoost outputs the final hypothesis h_f that is a weighted majority vote of T weak hypotheses where we assign the weight $-\ln \beta_t$ to h_t:

$$h_f(\mathbf{x}) = \begin{cases} 1 & \text{if } \sum_{t=1}^T (-\ln \beta_t) h_t(\mathbf{x}) \geq \sum_{t=1}^T (-\ln \beta_t) \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases}$$

Boosting Property

Since ϵ_t is less than $1/2$, there exists $0 < \gamma_t \leq 1/2$ such that $\epsilon_t = 1/2 - \gamma_t$. Observe that γ_t measures how much h_t's predictions are better than a hypothesis that guesses at random. Freund and Schapire proved the following theorem, which is called the *boosting property*:

Theorem 2.1 [10] Let ϵ denote the error of the final hypothesis; that is, $(\sum_{i=1}^N |h_f(\mathbf{x}_i) - y_i|)/N$. Then we have:

$$\epsilon \leq \Pi_{t=1}^T 2(\epsilon_t(1 - \epsilon_t))^{\frac{1}{2}} = \Pi_{t=1}^T (1 - 4\gamma_t^2)^{\frac{1}{2}}. \quad \blacksquare$$

The above theorem indicates that the error of the final hypothesis adapts to the error of individual weak hypothesis. If most weak hypotheses are

moderately accurate, the error of the final hypothesis drops exponentially fast, and hence the number of iterations T could be efficiently minimized.

3 Simple Classes of Weak Hypotheses

Decision Stumps

To design the weak learner of AdaBoost, we can flexibly choose any method for finding weak hypotheses. In the literature, C4.5 has been frequently used as a weak learner [3, 9]. In [9] Freund and Schapire employ C4.5 to generate two classes of weak hypotheses: large decision trees and single-test decision trees which are named "decision stumps." They remark that large decision trees are a kind of overly complex weak hypothesis and thereby fail to perform well against unseen test datasets. Decision stumps, on the other hand, are simple and hence may not be subject to overfitting to the training datasets. However, Domingo and Watanabe [7] reported that in later rounds of iterations in AdaBoost, the resulting decision stump is often too weak, and hence we have to generate a large number of very weak hypotheses for improving the prediction accuracy of the final hypothesis. To make matters worse, sometimes the weak learner cannot even output a sufficient number of weak hypotheses, because it cannot find any hypothesis whose error is smaller than 0.5. Once we encounter this situation, it is impossible to improve the final prediction accuracy. We will now present such an example.

Motivating Example

Figure 1 illustrates a sample training dataset of the following four hundred records:

$$\{(x_{i1}, x_{i2}, y_i) \mid x_{i1}, x_{i2} \in \{1, 2, \ldots, 20\}, y_i \text{ is either 0 or 1.}\}.$$

x_{i1} and x_{i2} are values of attributes A_1 and A_2 respectively. In Figure 1, black points indicate positive records ($y_i = 1$), while white points are negative ones ($y_i = 0$).

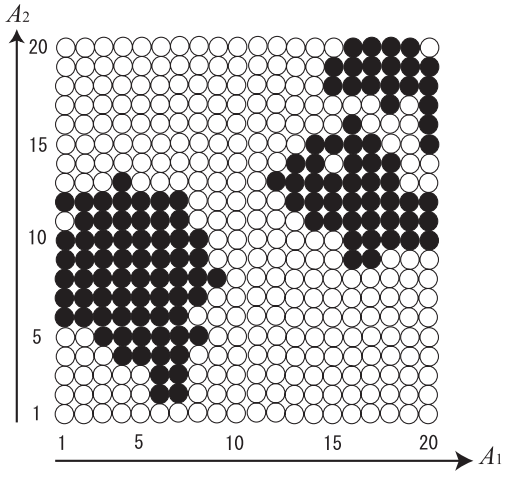

Figure 1: Motivating Example

Since the values in all records are numeric, it is natural to use decision stumps h_t defined by either

$$h_t((x_{i1}, x_{i2})) = 1 \text{ iff } v_1 \leq x_{i1}, \text{ or } h_t((x_{i1}, x_{i2})) = 1 \text{ iff } v_2 \leq x_{i2},$$

where $v_1, v_2 \in \{0, 1, 2, \ldots, 20\}$. We call those hypotheses *binary splitting*. For the purpose of improving the prediction accuracy of the final hypothesis, as Theorem 2.1 indicates, we should select the optimal binary splitting hypothesis that minimizes the error among all binary splitting hypotheses in

each round of AdaBoost. Actually it is not computationally costly to design such a weak learner, because we need to scan the domains of A_1 and A_2 just once. However, if we apply AdaBoost to the running dataset and iterate the generation of binary splitting hypotheses, the error of the eighteenth optimal hypothesis becomes 0.5, and hence we cannot further generate weak hypotheses. Figure 2 presents how the error of the final hypothesis changes if we increase the number of binary splitting hypotheses. The line graph terminates at the seventeenth hypothesis, because we could obtain only seventeen weak hypotheses.

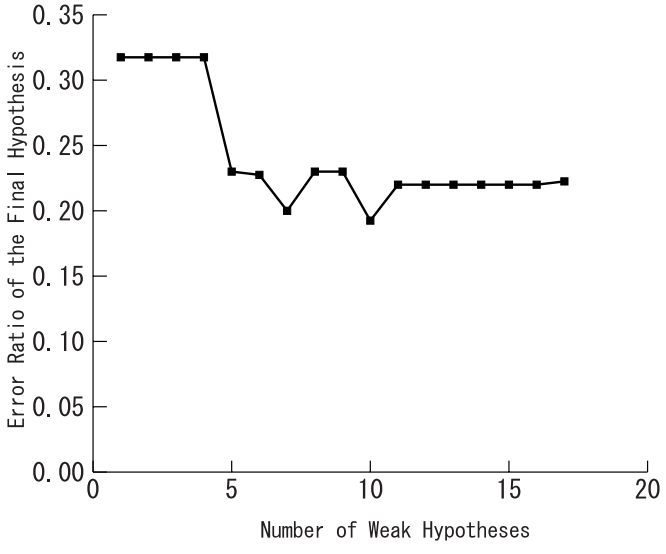

Figure 2: Error Curve of Using Binary Splitting Hypotheses

Simple but Expressive Hypotheses

It is natural to ask if there is an alternative class of weak hypotheses between the two extreme classes: large decision trees and simple decision stumps. Freund and Schapire [10] suggest that one plausible way is to restrict the weak learner to choose its weak hypotheses from some simple class of functions.

Following the line suggested, we consider several classes of simple but more expressive hypotheses that are broader than the class of decision stumps. These broader classes include ranges and regions for numeric attributes, subsets of categorical values, and conjunctions of Boolean tests. Searching such broader classes, we need to avoid selecting a very weak hypothesis from each class, and we expect to choose the optimal one that minimizes the error. However, it is a non-trivial question whether one can design an efficient algorithm for solving this optimization problem. Before presenting the solution to the optimization problem, we show what happens if we apply region splitting in replace of binary splitting to the running example.

Rectangular Region Splitting for Motivating Example

We here consider to use hypotheses h_t of the form:

$$h_t((x_{i1}, x_{i2})) = 1 \text{ iff } (x_{i1}, x_{i2}) \in [v_1, v_2] \times [w_1, w_2],$$

which we call *region splitting* hypotheses. In each step of AdaBoost, let us compute the optimal region splitting hypothesis that minimizes the error. Efficient algorithms for this purpose will be given in Section 4. Figure 3 presents how the error of the final hypothesis decreases as the number of region splitting hypotheses increases. The use of region splitting hypotheses outperforms the use of binary splitting hypotheses in many respects. First, we could generate more than 100 region splitting hypotheses. Second, the error of just one region splitting hypothesis is substantially smaller than that of using seventeen binary splitting hypotheses. Third, as Theorem 2.1 indicates, the error continues to decrease and reaches 0.0275 when one hundred region splitting hypotheses are used.

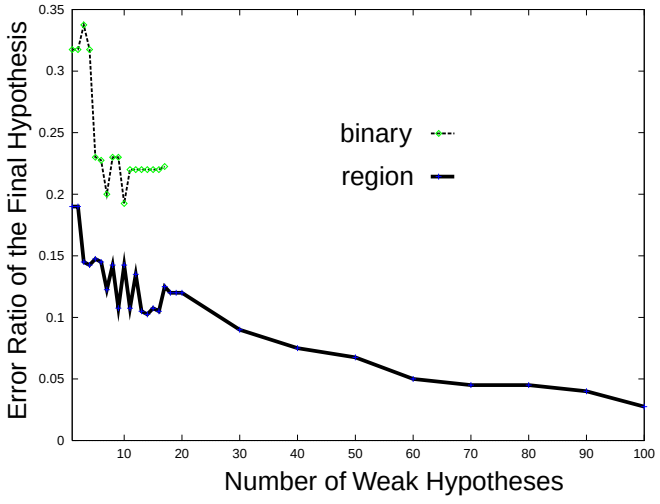

Figure 3: Error Curves: Region Splitting vs. Binary Splitting

Optimization Problem

Minimization of the error ϵ_t of the t-th weak hypothesis h_t is equivalent to maximization of the prediction accuracy, which is $1 - \epsilon_t$. Observe:

$$\begin{aligned}
1 - \epsilon_t &= \sum_{\{i|h_t(\mathbf{x}_i)=y_i\}} p_i^t \\
&= \sum_{\{i|h_t(\mathbf{x}_i)=y_i=1\}} p_i^t + \sum_{\{i|h_t(\mathbf{x}_i)=y_i=0\}} p_i^t \\
&= \sum_{\{i|h_t(\mathbf{x}_i)=y_i=1\}} p_i^t + (\sum_{\{i|y_i=0\}} p_i^t - \sum_{\{i|h_t(\mathbf{x}_i)=1,y_i=0\}} p_i^t) \\
&= (\sum_{\{i|h_t(\mathbf{x}_i)=1,y_i=1\}} p_i^t - \sum_{\{i|h_t(\mathbf{x}_i)=1,y_i=0\}} p_i^t) + \sum_{\{i|y_i=0\}} p_i^t.
\end{aligned}$$

Define

$$g_i^t = \begin{cases} p_i^t & \text{if } y_i = 1 \\ -p_i^t & \text{otherwise.} \end{cases}$$

Let us call g_i^t the *accuracy gain* (*gain*, for short) of $(\mathbf{x}_i, y_i)$ for h_t. Then,

$$1 - \epsilon_t = \sum_{\{i|h_t(\mathbf{x}_i)=1\}} g_i^t + \sum_{\{i|y_i=0\}} p_i^t.$$

Since $\sum_{\{i|y_i=0\}} p_i^t$ is independent of the choice of h_t, maximization of the acuuracy $1 - \epsilon_t$ is equivalent to maximization of $\sum_{\{i|h_t(\mathbf{x}_i)=1\}} g_i^t$. We call

$\sum_{\{i|h_t(\mathbf{x}_i)=1\}} g_i^t$ the *accuracy gain* (*gain*, for short) of h_t. In what follows, for each class of simple hypotheses, we present an efficient algorithm for choosing the optimal hypothesis whose gain are maximum in its class.

4 Optimal Ranges and Regions

We first introduce efficient algorithms for computing optimal ranges and then move on to optimal regions.

Optimal Ranges

Although $\mathbf{x}_i$ is a value vector, for simplicity, let us assume that $\mathbf{x}_i$ itself denotes a single numeric value. We use hypotheses h_t such that

$$h_t(\mathbf{x}_i) = 1 \ \text{ iff } \ \mathbf{x}_i \in [l, h].$$

Our goal is to compute an optimal range $[l, h]$ that maximizes the accuracy gain of h_t; namely

$$\max \sum_{\{i|\mathbf{x}_i \in [l,h]\}} g_i^t.$$

Without loss of generality, we assume that $\mathbf{x}_i$ are sorted in an ascending order $\mathbf{x}_1 \leq \mathbf{x}_2 \leq \mathbf{x}_3 \leq \ldots$. If some $\mathbf{x}_i, \ldots, \mathbf{x}_{i+k}$ are equal, we merge them in the sense that we take the sum of gains $g_i^t + \ldots + g_{i+k}^t$, assign the sum to $\mathbf{x}_i$, and rename the indexes so that all indexes are consecutive; namely, $\mathbf{x}_1 < \mathbf{x}_2 < \ldots$. Then, input the sequence of gains $g_1^t, g_2^t, \ldots$ to Kadena's algorithm [4]. Given a sequence of real numbers $g_1, g_2, \ldots, g_M$, Kadena's algorithm computes an optimal range $[s, t]$ that maximizes $\sum_{i \in [s,t]} g_i$ in $O(M)$-time.

It is natural to consider the use of more than one range for maximizing accuracy gain. In the last KDD conference, Brin, Rastogi, and Shim pre-

sented an efficient way of computing the optimal set of at most k ranges in $O(kM)$-time [5], which is a non-trivial extension of Kadena's algorithm.

Optimal Regions

In Section 3, we have discussed the advantage of using region splitting hypotheses of the form:

$$h_t((x_{i1}, x_{i2})) = 1 \text{ iff } (x_{i1}, x_{i2}) \in [v_1, v_2] \times [w_1, w_2].$$

We here present how to efficiently compute the optimal rectangle R that maximizes the accuracy gain; namely,

$$\max \sum \{g_i^t \mid (x_{i1}, x_{i2}) \in R\}.$$

In order to limit the number of rectangles to a moderate number, we first divide the domain of x_{i1} (also, x_{i2}) into M non-overlapping buckets such that their union is equal to the original domain. Using those buckets, we divide the two dimensional plane into M^2 pixels, and we represent a rectangle as a union of those pixels. Although the number of rectangles is $O(M^4)$, it is straightforward to design an $O(M^3)$-time algorithm by using Kadena's algorithm. The idea is that for each of ${}_MC_2$ pairs of rows, we apply Kadena's algorithm to calculate the optimal range of columns. Also, a subcubic time algorithm that uses funny matrix multiplication [14] is also available.

We are then interested in the design of efficient algorithm for computing more than one rectangles for maximizing the accuracy gain. However, Khanna, Muthukrishnan and Paterson remark that the problem is NP-hard [12]. Brin, Rastogi, and Shim present an approximation algorithm for this problem [5], however the approximation is within a factor of $\frac{1}{4}$ of the optimal solution. Thus computing the optimal set of more than one rectangles is

computationally intractable. In practice, however, calculating one rectangle in each round of AdaBoost would be sufficient to make a highly accurate prediction, as we have seen in the previous section. This is partly because the final hypothesis performs a weighted majority vote of weak hypotheses of rectangles, and this vote is almost equivalent to the task of checking the existence of a record in the union of these rectangles.

So far we have been focusing on rectangles. In general there have been developed efficient algorithms for computing the optimized gain region among various classes of two dimensional connected regions whose boundaries are more flexible than those of rectangles. For instance, an *x-monotone* region is such a connected region that the intersection with any column is undivided, and a *rectilinear convex* region is such an x-monotone region that the intersection with any row is also undivided. The optimized gain x-monotone (rectilinear convex, respectively) region can be computed in $O(M^2)$-time [11] (in $O(M^3)$-time [15]). The use of these regions is expected to further improve the prediction accuracy of the final hypothesis.

5 Optimal Conjunction

Given a dataset $\{(x_{i1}, x_{i2}, \ldots, x_{in}, y_i) \mid i = 1, \ldots, N\}$ such that x_{ij} is Boolean valued; that is, $x_{ij} \in \{0, 1\}$. In this section, we consider hypotheses h_t that are conjunctions of simple tests on each attribute; that is;

$$h_t((x_{i1}, x_{i2}, \ldots, x_{in})) = 1 \quad \text{iff} \quad x_{ij_1} = 1 \wedge \ldots \wedge x_{ij_M} = 1. \tag{1}$$

We are then interested in computing the optimal conjunction that maximizes the accuracy gain. The number of conjunctions of the form (1) is ${}_nC_M$. If we treat M as a variable, we can prove that the problem is NP-hard by

using the NP-completeness of the minimum set cover problem. The proof is length so we will omit it due to the space limitation. In practice, it would be reasonable to limit the number of conjunctions M to a small constant, say five. Then, the problem becomes to be tractable, but the problem still demands an efficient way of computing the optimal conjunction especially when the number of attributes n is fairly large.

Connection with Itemset Enumeration Problem

We first remark that the problem has a strong connection with itemset enumeration problem [1]. We then we will generalize the idea of Apriori algorithm [2] for computing the optimal conjunction efficiently.

Let $a_1, a_2, \ldots, a_n$ be n items. We will identify a record with an itemset according to the mapping ϕ:

$$\phi : (x_{i1}, \ldots, x_{in}) \ \rightarrow \ \{a_j \mid x_{ij} = 1\}.$$

For instance, $\phi((1,0,1,1,0)) = \{a_1, a_3, a_4\}$. We also regard the conjunction $x_{ij_1} = 1 \wedge \ldots \wedge x_{ij_M} = 1$ as $\{a_{j_1}, \ldots, a_{j_M}\}$. For example, we associate $x_{i3} = 1 \wedge x_{i4} = 1$ with $\{a_3, a_4\}$. We are able to rephrase the property that the record $(1,0,1,1,0)$ satisfies $x_{i3} = 1 \wedge x_{i4} = 1$ by using words of itemsets; namely, $\{a_1, a_3, a_4\} \supseteq \{a_3, a_4\}$. In general, we have the following equivalence:

$$\begin{aligned} & h_t((x_{i1}, x_{i2}, \ldots, x_{in})) = 1 \\ \text{iff} \quad & x_{ij_1} = 1 \wedge \ldots \wedge x_{ij_M} = 1 \\ \text{iff} \quad & \phi((x_{i1}, x_{i2}, \ldots, x_{in})) \supseteq \{a_{j_1}, \ldots, a_{j_M}\}. \end{aligned}$$

In what follows, for simplicity, let $\mathbf{x}_i$ denote $(x_{i1}, x_{i2}, \ldots, x_{in})$. Then, $\phi(\mathbf{x}_i)$ means $\phi((x_{i1}, x_{i2}, \ldots, x_{in}))$.

Now finding the optimal conjunction of the form (1) that maximizes the accuracy gain is equivalent to the computation of the itemset I that maximizes

$$\sum_{\{i|\phi(\mathbf{x}_i)\supseteq I\}} g_i^t.$$

Let us call the above sum of gains the *gain* of I. Let $gain(I)$ denote the sum. The gain of I is similar to the support of I, $|\{i \mid \phi(\mathbf{x}_i) \supseteq I\}|/N$, where N is the number of all records. Thus one may consider the possibility of applying the Apriori algorithm to the calculation of the optimal itemset I that maximizes $gain(I)$. To this end, however, Apriori needs a major conversion.

Extending the Idea of Apriori Algorithm

Let $support(I)$ denote the support of I. The support is *anti-monotone* with respect to set-inclusion of itemsets; that is, for any $J \supseteq I$, $support(J) \leq support(I)$. The Apriori algorithm uses this property to effectively prune away a substantial number of unproductive itemsets from its search space. However, the gain is not anti-monotone; namely, $J \supseteq I$ does not always imply $gain(J) \leq gain(I)$, because some g_i^t could be negative.

We solve this problem as follows: We scan the lattice of itemsets beginning with smaller itemsets and continuing to larger ones, according to Apriori's strategy. Suppose that we investigate an itemset I during the search. The following theorem presents a tight upper bound on $\{gain(J) \mid J \supseteq I\}$.

Theorem 5.1 For any $J \supseteq I$,

$$gain(J) \leq \sum_{\{i|\phi(\mathbf{x}_i)\supseteq I, y_i=1\}} g_i^t.$$

Proof: Recall

$$g_i^t = \begin{cases} p_i^t & \text{if } y_i = 1 \\ -p_i^t & \text{if } y_i = 0. \end{cases}$$

Then,

$$gain(I) = \sum_{\{i|\phi(\mathbf{x}_i)\supseteq I\}} g_i^t = \sum_{\{i|\phi(\mathbf{x}_i)\supseteq I, y_i=1\}} p_i^t - \sum_{\{i|\phi(\mathbf{x}_i)\supseteq I, y_i=0\}} p_i^t$$

Since $p_i^t \geq 0$, for each $v \in \{0, 1\}$,

$$\sum_{\{i|\phi(\mathbf{x}_i)\supseteq J, y_i=v\}} p_i^t \leq \sum_{\{i|\phi(\mathbf{x}_i)\supseteq I, y_i=v\}} p_i^t.$$

Consequently, $gain(J) \leq \sum_{\{i|\phi(\mathbf{x}_i)\supseteq I, y_i=1\}} g_i^t$. ∎

Definition 5.1 Let $u(I)$ denote the upper bound $\sum_{\{i|\phi(\mathbf{x}_i)\supseteq I, y_i=1\}} g_i^t$. ∎

During the scan of the itemset lattice, we always maintain the temporarily maximum gain among all the gains calculated so far and set it to τ. If $u(I) < \tau$, no superset of I gives a gain greater than or equal to τ, and hence we can safely prune all supersets of I at once. On the other hand, if $u(I) \geq \tau$, I is promising in the sense that there might exist a superset J of I such that $gain(J) \geq \tau$.

Definition 5.2 Suppose that τ is given and fixed. An itemset is a k-itemset if it contains exactly k items. An itemset I is promising if $u(I) \geq \tau$. Let P_k denote the set of promising k-itemsets. ∎

Thus we will search $P_1 \cup P_2 \cup \ldots$ for the optimal itemset. Next, to accelerate the generation of P_k, we introduce a candidate set for P_k.

Definition 5.3 An itemset I is *potentially promising* if every proper subset of I is promising. Let Q_k denote the set of all potentially promising k-itemsets. ∎

The following theorem guarantees that Q_k is be a candidate set for P_k.

Theorem 5.2 $Q_k \supseteq P_k$. ∎

$\tau := 0$;
$Q_1 := \{I \mid I \text{ is a 1-itemset.}\}$; $k := 1$;
repeat begin
    If $k > 1$, generate Q_k from P_{k-1};
    For each $I \in Q_k$, scan all the records to compute $u(I)$ and $gain(I)$;
    $\tau := \max(\tau,\ \max\{gain(I) \mid I \in Q_k\})$;
    $P_k := \{I \in Q_k \mid u(I) \geq \tau\}$; $X := P_k$; $k++$;
end until $X = \phi$;
Return τ with its corresponding itemset;

Figure 4: AprioriGain for Computing the Optimal Itemset

The benefit of Q_k is that Q_k can be obtained from P_{k-1} without scanning all records that may reside in the secondary disk. To this end, we use the idea of the `apriori-gen` function of the Apriori algorithm [2]; that is, we select two members in P_{k-1}, say I_1 and I_2, such that I_1 and I_2 share $(k-2)$ items in common, and then check to see whether each $(k-1)$-itemset included in $I_1 \cup I_2$ belongs to P_{k-1}, which can be determined efficiently by organizing P_{k-1} as a hash tree structure. We repeat this process to create Q_k. Figure 4 presents the overall algorithm, which we call AprioriGain (Apriori for optimizing Gain).

Performance Result

We have applied AprioriGain to datasets that were generated using the method introduced by Agrawal and Srikant [2]. The execution time of Apri-

oriGain scaled almost linearly with the number of records. Due to the space limitation, we omit the performance results, which will be presented in the full version of this paper.

6 Optimal Subset of Categorical Values

In this section, for simplicity, let us assume that $\mathbf{x}_i$ itself denotes a single categorical value. Let $\{c_1, c_2, \ldots, c_M\}$ be the domain of the categorical attribute. Typical hypotheses h_t would be of the form:

$$h_t(\mathbf{x}_i) = 1 \text{ iff } \mathbf{x}_i = c_j.$$

Computing the optimal choice of c_j that maximizes the accuracy gain is inexpensive. In practice, the number of categorical values M could be fairly large; for instance, consider the number of countries in the world. In such cases, the number of records satisfying $\mathbf{x}_i = c_j$ could be relatively small, thereby raising the error of the hypothesis h_t. One way to overcome this problem is to use a subset S of $\{c_1, c_2, \ldots, c_M\}$ instead of a single value and to employ hypotheses of the form:

$$h_t(\mathbf{x}_i) = 1 \text{ iff } \mathbf{x}_i \in S.$$

Our goal is then to find S that maximizes the sum of gains $\sum_{\mathbf{x}_i \in S} g_i^t$. Although the number of possible subsets of $\{c_1, c_2, \ldots, c_M\}$ is 2^M, we are able to compute the optimal subset S in $O(M)$-time. First, without loss of generality, we assume that

$$\sum_{\{i|\mathbf{x}_i=c_1\}} g_i^t \geq \sum_{\{i|\mathbf{x}_i=c_2\}} g_i^t \geq \ldots \geq \sum_{\{i|\mathbf{x}_i=c_M\}} g_i^t.$$

Otherwise, we rename the indexes so that the above property is guaranteed. We then obtain the following property.

Theorem 6.1 Let

$$S = \{c_j \mid \sum_{\{i|\mathbf{x}_i = c_j\}} g_i^t \geq 0\}.$$

S maximizes $\sum_{\mathbf{x}_i \in S} g_i^t$. ∎

Thus we only need to find the maximum index k such that

$$\sum_{\{i|\mathbf{x}_i = c_k\}} g_i^t \geq 0,$$

returning $S = \{c_j \mid j = 1, \ldots, k\}$ as the answer. Consequently, the optimal subset can be computed in $O(M)$-time.

7 Discussion

To improve the prediction accuracy of AdaBoost, we have presented efficient algorithms for several classes of simple but expressive hypotheses. In the literature, boosting algorithms have been developed in the machine learning community, while optimization algorithms for association rules and optimized ranges/regions have been proposed and studied in the database and data mining communities. This paper sheds light on a strong connection between AdaBoost and optimization algorithms for data mining.

There are some interesting open problems regarding the design of optimization algorithms. For instance, let D_1 and D_2 be the domains of two categorical attributes. The question is if we can develop an efficient algorithm for computing a pair of $S_1 \subset D_1$ and $S_2 \subset D_2$ that maximizes

$$\sum \{g_i^t \mid x_{i1} \in S_1, x_{i2} \in S_2\}.$$

Incidentally many researchers have been evaluating the empirical performance of AdaBoost by using decision trees or decision stumps as weak hypotheses [6, 9]. For instance, an elaborate analysis can be found in [6]. We

plan to perform empirical tests by using optimal hypotheses presented in this paper.

Acknowledgement We thank Carlos Domingo, Naoki Katoh, and Osamu Watanabe for their valuable input.

References

[1] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993*, pages 207–216. ACM Press, 1993.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499. Morgan Kaufmann, 1994.

[3] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(2):105–139, 1999.

[4] J. Bentley. Programming pearls. *Communications of the ACM*, 27(27):865–871, Sept. 1984.

[5] S. Brin, R. Rastogi, and K. Shim. Mining optimized gain rules for numeric attributes. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 15-18 August 1999, San Diego, CA USA*, pages 135–144. ACM Press, 1999.

[6] T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning (to appear)*, `http://www.cs.orst.edu/ tgd/cv/pubs.html`.

[7] C. Domingo and O. Watanabe. A modification of adaboost: A preliminary report. *Research Reports, Dept. of Math. and Comp. Sciences, Tokyo Institute of Technology*, (C-133), July 1999.

[8] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

[9] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156, 1996.

[10] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, Aug. 1997.

[11] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Data mining using two-dimensional optimized accociation rules: Scheme, algorithms, and visualization. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*, pages 13–23. ACM Press, 1996.

[12] S. Khanna, S. Muthukrishnan, and M. Paterson. On approximating rectangle tiling and packing. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 384–393, Jan. 1998.

[13] R. E. Schapire. The strength of weak learnability (extended abstract). In *FOCS*, pages 28–33, 1989.

[14] H. Tamaki and T. Tokuyama. Algorithms for the maxium subarray problem based on matrix multiplication. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 446–452, Jan. 1998.

[15] K. Yoda, T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Computing optimized rectilinear regions for association rules. In *Proceedings of the*

Third International Conference on Knowledge Discovery and Data Mining, pages 96–103, Aug. 1997.